# Integration Manual –NvMMgr

Table of Contents

[1 Dependencies 2](#_Toc357692818)

[1.1 SWCs 2](#_Toc357692819)

[1.2 Functions to be provided to Integration Project 2](#_Toc357692820)

[2 Configuration 3](#_Toc357692821)

[2.1 Build Time Config 3](#_Toc357692822)

[2.2 Configuration Files to be provided by Integration Project 3](#_Toc357692823)

[2.2.1 Da Vinci Config generation 3](#_Toc357692824)

[2.2.2 Manual Configuration Changes 3](#_Toc357692825)

[3 Integration 4](#_Toc357692826)

[3.1 Required Global Data Inputs 4](#_Toc357692827)

[3.2 Optional Global Data Inputs 4](#_Toc357692828)

[3.3 Specific Include Path present 4](#_Toc357692829)

[4 Runnable Scheduling 5](#_Toc357692830)

[5 Memory Mapping 6](#_Toc357692831)

[5.1 Mapping 6](#_Toc357692832)

[5.2 Usage 6](#_Toc357692833)

[5.3 NvM Blocks 6](#_Toc357692834)

[6 Compiler Settings 6](#_Toc357692835)

[6.1 Preprocessor MACRO 6](#_Toc357692836)

[6.2 Optimization Settings 6](#_Toc357692837)

[7 Revision Control Log 7](#_Toc357692838)

# Dependencies

## SWCs

|  |  |
| --- | --- |
| Module | Required Feature |
| NvM | Entire BSW |
| MemIf | Entire BSW |

Note : Referencing the external components should be avoided in most cases. Only in unavoidable circumstance external components should be refered. Developer should track the references.

## Global Functions(Non RTE) to be provided to Integration Project

### FeeIf

There are differing functions provided by this module depending on how it is configured through configurator. These functions are listed below based on the generated configuration. The main idea is that if the configuration is going to require configuring trusted function calls through the O/S because of the requirement that the fee driver must execute in a privileged/trusted mode, the wrapper functions to implement the trusted functions will be provided by this module automatically. The notes indicate the intended caller of the function (be it the integrator, another BSW (O/S or MemIf), or if the function is strictly called internally by this module).

#### Functions provided if BC\_FEEIF\_ECUSTARTUPTRUSTED == STD\_OFF

* FeeIf\_Init (for NvMMgr internal use only)
* TWrapC\_FeeIf\_Init (for integrator scheduling)
* TRUSTED\_TWrapS\_FeeIf\_Init (for O/S)
* TWrapC\_Fee\_MainFunction (for integrator scheduling)
* TRUSTED\_TWrapS\_Fee\_MainFunction (for O/S)

#### Functions provided if BC\_FEEIF\_ECUSTARTUPTRUSTED == STD\_ON

* FeeIf\_Init (for integrator scheduling)

#### Functions provided if BC\_FEEIF\_NVMTRUSTED == STD\_OFF

* TWrapC\_Fee\_Read (for MemIf)
* TRUSTED\_TWrapS\_Fee\_Read (for O/S)
* TWrapC\_Fee\_Write (for MemIf)
* TRUSTED\_TWrapS\_Fee\_Write (for O/S)
* TWrapC\_Fee\_EraseImmediateBlock (for MemIf)
* TRUSTED\_TWrapS\_Fee\_EraseImmediateBlock (for O/S)
* TWrapC\_Fee\_InvalidateBlock (for MemIf)
* TRUSTED\_TWrapS\_Fee\_InvalidateBlock (for O/S)
* TWrapC\_Fee\_Cancel (for MemIf)
* TRUSTED\_TWrapS\_Fee\_Cancel (for O/S)
* TWrapC\_Fee\_GetStatus (for MemIf)
* TRUSTED\_TWrapS\_Fee\_GetStatus( for O/S)
* TWrapC\_Fee\_GetJobResult(for MemIf)
* TRUSTED\_TWrapS\_Fee\_GetJobResult (for O/S)
* TWrapC\_TI\_Fee\_SuspendResumeErase (for MemIf)
* TRUSTED\_TWrapS\_TI\_Fee\_SuspendResumeErase (for O/S)

### Fee Subproject

Please refer to BSW AUTOSAR specification. Integrator only responsible for scheduling Fee\_MainFunction(). Other APIs are used by NvM/MemIf BSWs and FeeIf.

### Fls Subproject

APIs not typically used in integration project except by Fee BSW. Please refer to TI documentation for API descriptions.

# Configuration

## Build Time Config

|  |  |  |
| --- | --- | --- |
| Modules | Notes |  |
| None |  |  |

## Configuration Files to be provided by Integration Project

<Configuration file that will generated from this components that will require Da Vinci Config generation or manual generation. Describe each parameter >

### Da Vinci Parameter Configuration Changes

|  |  |  |
| --- | --- | --- |
| Parameter | Notes | SWC |
| /NvMMgr/NvMMrGeneral/EcuStartupApp | Integrator to choose the application that the EcuStartup Init2 routine will execute from. This will allow the generator to determine the value of BC\_FEEIF\_ECUSTARTUPTRUSTED | NvMMgr |
| /NvMMgr/NvMMrGeneral/NvMandMemIfApp | Integrator to choose the application that the NvM and MemIf functions will execute from (NvM\_MainFunction). This will allow the generator to determine the value of BC\_FEEIF\_NVMTRUSTED | NvMMgr |
| /NvM/NvmCommon/NvmPollingMode | Must be set to “True” since current Fee Driver only supports Polling Mode | NvM |
| /NvM/Nvm\_30\_CommonVendorParams/NvmKillWriteAllApi | Must be set to “False” if BC\_FEEIF\_NVMTRUSTED == STD\_OFF for current Vector NvM delivered BSW. This is required since there is a critical section that wraps around the call to the FEE driver WriteBlock API in NvM if the KillWriteAllApi is turned on. The current strategy of using trusted function calls for the FEE APIs is not allowed by the O/S within a critical section that is disabling all interrupts. If the KillWriteAllApi is required in the future, a different critical section handling strategy may need to be investigated for the NvM driver. | NvM |
| /Fee/FeeGeneral/FeeDevErrorDetect | Determines if Fee makes use of DET module | Fee |
| /Fee/FeeGeneral/FeeCRCEnable | This should be set to “False” for Nexteer use case. If set to “True” a Fletcher checksum will be used to determine if the data to be written to FEE is the same as the data currently in FEE memory. Because different data can result in the same checksum, this opens up an opportunity for writes to be lost. Setting this to “False” will cause Fee to do a direct byte-by-byte comparison of the data instead of using the checksum. | Fee |
| /Fee/FeeGeneral/FeeWriteCounterSave | This should be set to “False” for Nexteer use case. This enables writing a counter for each block to indicate how many times it was written. Currently, the Fee driver doesn’t use this data and it is not accessible through the API so overhead of writing provides no benefit currently. | Fee |
| /Fee/FeeGeneral/FeeNumberOfUnconfiguredBlocksToCopy | This parameter is used to tell the Fee Driver how many blocks it should support copying during a virtual sector swap that are not part of the Fee configuration. There are two main uses of this parameter. One is to allow compatibility when flashing different version of application software with differing NvM maps. Another is to allow a bootloader to have a reduced Fee configuration compared to the application (and not having to update the bootloader every time the application NvM map changes). An application should have a large enough value to support compatibility with different application versions (so it depends on how close to a final production NvM map is integrated). The closer to the final NvM map, the smaller this number can be. A bootloader should have a large value in this parameter (larger than the number of blocks expected in the application at start of production). | Fee |
| /Fee/FeeGeneral/FeeNumberOfEightByteWrites | Indicates to the Fee Driver how many sets of 8 bytes are written per Fee mainfunction cycle. Typical Nexteer setting is 128 (allowing 1024 bytes of data per mainfunction loop). From timing analysis, writing this much data was still under the timing the mainfunction took when Fee was doing a virtual sector swap. It would be preferable that this parameter is at least as large as the largest size block configured. | Fee |
| /Fee/FeeSectorConfiguration# | See the following picture for typical Nexteer usage:  SectorConfig.TIF | Fee |
| /Fee/FeeBlockConfiguration# | Integration project specific. See notes in Integration section. | Fee |
| /Fee/FeeGeneral/FeeNvmJobEndNotification | Should be set to NvM\_JobEndNotification, however, unused currently since Fee Driver only supports NvM in polling mode | Fee |
| /Fee/FeeGeneral/FeeNvmJobErrorNotification | Should be set to NvM\_JobErrorNotification, however, unused currently since Fee Driver only supports NvM in polling mode | Fee |
| /Fee/FeeGeneral/FeeFrequency | Set to the clock frequency (in MHz) that the uController is running at. Typical value of 160 | Fee |
| /Fee/FeeGeneral/FeePollingMode | Must be set to “True” since current Fee Driver only supports Polling Mode | Fee |
| /Fee/FeeGeneral/FeeEnableErrorCorrection | Typically set to “False” for Nexteer use – not currently used at all by the Fee Driver. Please note that single bit ECC data correction will still occur even if this setting is false. | Fee |
| /Fee/FeeTypes/FeeFlashErrCorrHandlingType | Typically set to “None” for Nexteer use since critical NvM data is validated by other mechanisms. Please note that single bit ECC data correction will still occur even if this setting is false. | Fee |
| /Fee/FeeGeneral/FeeNumberofEEPS | Typically set to 2 for Nexteer use to enable support of two independent EEPROMs. In a bootloader it may be desired to only setup one so the bootloader doesn’t have access to the other. | Fee |
| /Fee/FeeGeneral/FeeNumberOfVirtualSectorsEEP1 | Typically set to 2 for Nexteer use to enable support of two independent EEPROMs. Note this is only required if there is more than one EEPROM configured (see above). | Fee |
| /Fee/FeeGeneral/FeeVersionInfoApi | Typically set to “False” for Nexteer usage since currently the Version number is not used | Fee |

### DaVinci Interrupt Configuration Changes

|  |  |  |  |
| --- | --- | --- | --- |
| ISR Name | VIM # | Priority Dependency | Notes |
| None |  |  |  |

### Manual Configuration Changes

|  |  |  |
| --- | --- | --- |
| Constant | Notes | SWC |
| None |  |  |

# Integration

**General Notes and O/S Configuration for Trusted Functions:**

Please note that this module currently only contains the Fee related files of the AUTOSAR memory stack. It is assumed that NvM and MemIf components are already integrated. The entire Fee and Fls modules must be configured to be run in a privileged/trusted mode on the uController. Because of this and depending on the configuration of the project being integrated into, trusted functions may be required to be setup in the O/S. The following tables show which trusted functions need to be configured in the O/S in the application that Fee and Fls will reside in. The Cd\_NvMMgr\_Cfg.h file that is generated by configurator will show the integration project build constant settings (BC\_FEEIF\_ECUSTARTUPTRUSTED, BC\_FEEIF\_NVMTRUSTED) that are referenced below.

|  |  |  |
| --- | --- | --- |
| Trusted Function Name | Passed Parameters | Return Type |
| Trusted Functions to Setup In O/S if BC\_FEEIF\_ECUSTARTUPTRUSTED == STD\_OFF | | |
| TWrapS\_FeeIf\_Init | **void** | **void** |
| TWrapS\_Fee\_MainFunction | **void** | **void** |
| Trusted Functions to Setup In O/S if BC\_FEEIF\_NVMTRUSTED == STD\_OFF | | |
| TWrapS\_Fee\_Read | **uint16 BlockNumber,uint16 BlockOffset,uint8\* DataBufferPtr,uint16 Length** | **Std\_ReturnType** |
| TWrapS\_Fee\_Write | **uint16 BlockNumber,uint8\* DataBufferPtr** | **Std\_ReturnType** |
| TWrapS\_Fee\_EraseImmediateBlock | **uint16 BlockNumber** | **Std\_ReturnType** |
| TWrapS\_Fee\_InvalidateBlock | **uint16 BlockNumber** | **Std\_ReturnType** |
| TWrapS\_Fee\_Cancel | **void** | **void** |
| TWrapS\_Fee\_GetStatus | **void** | **uint8Note1** |
| TWrapS\_Fee\_GetJobResult | **void** | **uint8Note1** |
| TWrapS\_TI\_Fee\_SuspendResumeErase | **uint8 Note2** | **void** |

Note1 - Please note that the actual return type of Fee\_GetStatus is an enumeration of type MemIf\_StatusType and the actual return type of Fee\_GetJobResult is an enumeration of type MemIf\_JobResultType. Due to a current limitation of the include path control of trusted functions in the O/S, the trusted functions are setup with standard uint8 return types and appropriate typecasting is used where needed. This is acceptable under the assumption that the enumerations will resolve into an underlying uint8 type on the target. This workaround can be removed from the component if in the future the O/S provides a better mechanism for header inclusion in the trusted functions.

Note2 – Similar to Note1, the data type passed into the SuspendResumeErase function is an enumerated value TI\_Fee\_EraseCommandType. The input is type casted in the Cd\_FeeIf.c function call to the correct type.

**TI\_Fee\_SuspendResumeErase:**

Please see the TI documentation for details on when and how to use this feature.

|  |
| --- |
| **--- IMPORTANT ---** |
| The files for Configurator do not currently support the latest Fee driver. The integrator will need to modify the generated Fee\_Cfg.h file and create a definition for FEE\_TOTAL\_BLOCKS\_DATASETS.  This value is calculated by summing the number of data sets for each NvM block. For example, if a configuration contained three NvM entries, one entry was a native block (Number of data sets = 1), one entry was a redundant block (Number of data sets = 2), and the final block was a data set with the number of sets set to 10. The value for FEE\_TOTAL\_BLOCKS\_DATASETS would be (1) + (2) + (10) or 13.  This value should be added as a #define at the end of the Fee\_Cfg.h.  Once the updated Configurator files have been delivered and integrated into the component baseline, this table can be removed. |

**Fapi\_UserDefinedFunctions.c:**

This file provides a function that is required for the TI flash library (FLS subproject). It is technically supposed to be user-configurable based on if special watchdog servicing code is required during the execution of the flash library functions; however, Nexteer’s typical use case does not need special handling, so an empty stub function is provided in the NvMMgr component for integration.

**NvM BSW Configuration:**

Please note that the current Fee driver REQUIRES NvM to be placed into “Polling Mode”. Please see NvM configuration parameter (/NvM/NvmCommon/NvmPollingMode).

**Special MemMap MemIf Settings if BC\_FEEIF\_NVMTRUSTED == STD\_OFF:**

Since the Fee driver requires to be run in a privileged/trusted mode, the calls into Fee must be intercepted by the O/S and replaced by trusted function calls to put the uController into a privileged mode before executing any Fee functions. Unfortunately, the current MemIf BSW does not allow user-configurable Fee API function calls to allow this. A workaround has been developed by macro-replacing the Fee APIs with the trusted function APIs in the MemMap statement around the MemIf Fee API function pointer table. The following is a reference that can be used to apply this workaround in the integration project MemMap.h file:

#ifdef MEMIF\_START\_SEC\_CONST\_32BIT

#undef MEMIF\_START\_SEC\_CONST\_32BIT

#include "Cd\_FeeIf.h"

#define START\_SEC\_CONST\_32BIT

#define Fee\_Read TWrapC\_Fee\_Read

#define Fee\_Write TWrapC\_Fee\_Write

#define Fee\_EraseImmediateBlock TWrapC\_Fee\_EraseImmediateBlock

#define Fee\_InvalidateBlock TWrapC\_Fee\_InvalidateBlock

#define Fee\_Cancel TWrapC\_Fee\_Cancel

#define Fee\_GetStatus (MemIf\_ApiGetJobResultType)TWrapC\_Fee\_GetStatus

#define Fee\_GetJobResult (MemIf\_ApiGetJobResultType)TWrapC\_Fee\_GetJobResult

#endif

#ifdef MEMIF\_STOP\_SEC\_CONST\_32BIT

#undef MEMIF\_STOP\_SEC\_CONST\_32BIT

#define STOP\_SEC\_CONST

#undef Fee\_Read

#undef Fee\_Write

#undef Fee\_EraseImmediateBlock

#undef Fee\_InvalidateBlock

#undef Fee\_Cancel

#undef Fee\_GetStatus

#undef Fee\_GetJobResult

#endif

**Fee EEPROM strategy and Block Configuration Notes:**

The Fee Driver allows configuration of two independent EEPROMs. Nexteer’s strategy for usage of these two EEPROMs is to use one (FEE0) for data that is written once or very infrequently and to use one (FEE1) for data written frequently. The following shows some general guidelines for what data should fall into each FEE:

* FEE0:
  + Most data written during Nexteer or Customer Manufacturing process (cals, partnumbers)
  + Bootloader and bootloader/application shared data
* FEE1:
  + TypeH Data
  + Learned algorithmic data
  + DTC/NTC Fault Information
  + Data updated every ignition cycle (e.g. Ignition Counter, EEPROM close check)
  + Nexteer/Customer MEC

In FEE0, all of the bootloader required NvM blocks should be configured starting at block number 1 and be continuously sequentially incrementing. This forces the requirement that all bootloader data must be defined correctly for the first bootloader release with Fee integrated (or else compatibility issues will have to be considered).

Additionally, once a Fee block is defined in an application, it should never be removed. If the block becomes removed and is no longer needed, it should become a deprecated block. This strategy will ensure compatibility between software versions. **Removal of configured blocks should be done only if software compatibility is not a concern as determined by program consensus!!**

**CCS Global Settings:**

Code Composer Studio automatically searches all project folders for .c files to be included in a build. The Fee subproject has some vector template files in the “generate” folder that have an extension of .c and .h that are encrypted and should not be part of this build. To work around this, it is suggested to globally apply a filter to exclude all folders called “generate” from the build in code composer project settings (note this approach assumes ALL generate folders do not contain files required as part of the build). This can be done in the CCS project properties “Resource Filters”.

## Required Global Data Inputs

None

## Required Global Data Outputs

None

## Specific Include Path present

Yes (including for Fee and Fls sub-projects)

# Runnable Scheduling

This section specifies the required runnable scheduling. It is divided showing the requirements based on the configuration of the module.

## Integrator Common Scheduling in all configurations

|  |  |  |
| --- | --- | --- |
| Runnable | Scheduling Requirements | Trigger |
| Fee\_MainFunction | Scheduled in SchM task periodically. Must be run from a trusted application. Note it is highly recommended that this function run in the lowest priority task (just above the background task) since it can take a long time to execute during virtual sector swapping. | SchM 100ms typical |

## Integrator Scheduling if BC\_FEEIF\_ECUSTARTUPTRUSTED == STD\_OFF

|  |  |  |
| --- | --- | --- |
| Init | Scheduling Requirements | Trigger |
| TWrapC\_FeeIf\_Init | Executed in EcuStartup Init2 (after O/S starts but before RTE start) prior to NvM Upload | Once prior to RTE Start |

|  |  |  |
| --- | --- | --- |
| Runnable | Scheduling Requirements | Trigger |
| TWrapC\_Fee\_MainFunction | In a tight loop during NvM upload process after the NvM\_MainFunction (typically in Appl\_WaitNvMReady()) | N/A |

## Integrator Scheduling if BC\_FEEIF\_ECUSTARTUPTRUSTED == STD\_ON

|  |  |  |
| --- | --- | --- |
| Init | Scheduling Requirements | Trigger |
| FeeIf\_Init | Executed in EcuStartup\_Init2 (after O/S starts but before RTE start) prior to NvM Upload | Once prior to RTE Start |

|  |  |  |
| --- | --- | --- |
| Runnable | Scheduling Requirements | Trigger |
| Fee\_MainFunction | In a tight loop during NvM upload process after the NvM\_MainFunction (typically in Appl\_WaitNvMReady()) | N/A |

**.**

# Memory Mapping

## Mapping

|  |  |  |
| --- | --- | --- |
| Memory Section | Contents | Notes |
| FEE\_START\_SEC\_CODE |  | Please note that a VAR section must be opened and closed for this section for placement of function level static variables. The VAR section must be mapped to a trusted application. |
| FEE\_START\_SEC\_CONST\_UNSPECIFIED |  |  |
| FEE\_START\_SEC\_VAR\_INIT\_UNSPECIFIED |  | Must be mapped to a trusted application. |
| F021\_API\_CortexR4\_BE\_V3D16.lib |  | All RAM for this library must be mapped to a trusted application. |

\* Each …START\_SEC… constant is terminated by a …STOP\_SEC… constant as specified in the AUTOSAR Memory Mapping requirements.

## Usage

|  |  |  |
| --- | --- | --- |
| Feature | RAM | ROM |
| N/A |  |  |

Table 1: ARM Cortex R4 Memory Usage

## Non RTE NvM Blocks

|  |
| --- |
| Block Name |
| N/A |

Note : Size of the NVM block if configured in developer

## RTE NvM Blocks

|  |
| --- |
| Block Name |
| N/A |

Note : Size of the NVM block if configured in developer

# Compiler Settings

## Preprocessor MACRO

None

## Optimization Settings

None

# Revision Control Log

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev #** | **Change Description** | **Date** | **Author** |
| 1 | Initial version | 07/11/13 | LWW |
| 2 | Added note about running Fee\_MainFunction in a very low priority task | 07/25/13 | LWW |
| 3 | Added note about KillWriteAllApi configuration setting requirement in NvM driver | 08/02/13 | LWW |
| 4 | Added information about Suspend/Resume erase APIs and also the patch that is required for Fee\_Cfg.h until the new Configurator generator is integrated. | 11/04/14 | KJS |
| 5 | Added FEE Single bit ECC correction | 12/15/14 | PS |
| 6 | Updated FEE component and removed Nexteer FEE ECC workaround | 1/27/15 | PS |
| 7 | Updated FEE component to version 03.01.00\_01.23.00 | 10/15/2015 | PS |
| 8 | Updated FEE component to version 03.01.00\_01.23.01 for anomaly EA3#3993 correction | 02/10/2016 | KJS |